

3DPX - An Open-Source Methodology for 3D Physical Design Exploration

George Goudroumanis[†], Maria Pantazi-Kypraiou[†], George Floros[§],
Athanasios Tziouvaras[†], George Stamoulis[†], Alberto García-Ortiz^{*}

[†]Department of Electrical and Computer Engineering, University of Thessaly, Greece

[§]Department of Electronic and Electrical Engineering, Trinity College Dublin, Ireland

^{*}Institute of Electrodynamics and Microelectronics, University of Bremen, Germany

{gggeorgios-r, mpantazi-, attziouv, georges}@e-ce.uth.gr, florosg@tcd.ie, agarcia@item.uni-bremen.de

Abstract—Architectural exploration of novel technological options, such as 3D integration, requires close interaction with physical implementation. However, the lack of information regarding the technical aspects of the 3D flavor, and lack of open source tools are the key obstacles inhibiting the wide use of physical-aware 3D architectural exploration. To palliate this problem, we present 3DPX, an open-source methodology for 3D physical design exploration based on the OpenROAD framework. By leveraging open standards and tools, our methodology enables evaluation of the impact of 3D stacking on performance, power, and area (PPA). Experimental results on a set of RISC-V benchmark circuits show the expected 50% area reduction, while allowing users to analyze and optimize timing and power just as they would in a traditional 2D flow.

I. INTRODUCTION

As semiconductor scaling approaches physical limits, 3D ICs have emerged as a promising alternative. 3D ICs offer significant advantages in performance, power efficiency, and form factor by vertically stacking multiple dies [1]. While some dedicated EDA tools for 3D exist, their performance compared to the 2D alternatives is still insufficient. Therefore, a plethora of pseudo-3D flows using commercial 2D tools have emerged. However, open-source alternatives are lacking, limiting accessibility for academic and industrial research, especially in the domains of architectural exploration.

More specifically, industrial tools require early commitment to fixed partitioning schemes and proprietary technologies, while academic solutions address only isolated aspects of the design process, such as placement or routing, in isolation [2]. Even open-source alternatives, such as Open3DBench [3] impose restrictive assumptions, since they limit tier configurations to memory-on-logic stacks. On the other hand, more general methodologies as Hier-3D [4] depend on commercial toolchains. This critical gap in early-stage design space exploration forces designers to either accept significant area and power overheads or risk costly redesign cycles.

To address these challenges, we present the first complete open-source design space exploration 3D IC flow built based on OpenROAD [5]. Our framework, 3DPX², integrates end-to-end support for partitioning, placement, routing, and timing analysis exploration across multiple dies, while maintaining compatibility with diverse bonding configurations, as face-to-face (F2F), face-to-back (F2B), and monolithic. Using OpenROAD's modular infrastructure, we extend its 2D capabilities

to support hierarchical 3D designs, enabling logic-on-logic and memory-on-logic integration with multi-PDK support. This allows designers to evaluate trade-offs in performance, power, and area (PPA) across technology nodes and stacking methodologies. The main contributions of this paper are summarized hereafter. *First*, we provide a design space exploration framework that assists architectural evaluation of different partitioning schemes and bonding configurations. *Second*, the multi-PDK compatibility enables analysis and benchmarking across different technology nodes and bonding approaches. *Third*, our hierarchical implementation methodology is able to support large industrial designs, as well as both memory-on-logic and logic-on-logic architectures across multiple dies. Experimental results on a set of benchmark circuits endorse our claim about the design exploration nature of this flow.

II. 3DPX - PROPOSED FLOW

To provide an overview of the proposed methodology, we begin by illustrating its key components and workflow in Fig. 1. In the diagram, grey shapes represent user inputs and the orange and blue rectangles represent the PDK preparation step, and the top level and hierarchical ASIC flow, respectively. The goal of the PDK preparation step is to either extend an existing 2D PDK into a homogeneous 3D version or combine multiple 2D PDKs into a heterogeneous 3D PDK. In the second step, we introduce a hierarchical place-and-route (P&R) flow to facilitate the design exploration of large-scale systems. The design flow is entirely based on OpenROAD, complemented by our Python- and TCL-based API, which provides the necessary modularity and flexibility to the designer. To avoid confusion, it is important to clarify that, due to the lack of native hierarchical flow in OpenROAD, we managed to exploit this behavior by leveraging the LEF/DEF files native support. This way, combining our TCL scripts API and the LEF/DEF files support, we can temporarily store the P&R data of each hierarchical block, and merge them together using our Python interface, effectively creating a complete hierarchical flow.

1) *PDK preparation flow*: The first part begins with the LEF file modification. To extend a LEF file to be used in a 3D monolithic ASIC flow, we have to extend the layer stack to the number of tiers we want to incorporate for our 3D IC. Because the metal layers' ordering in the LEF file determines the physical order of the layers in the chip, we have to select the 3D paradigm we want to move on with first. Also, if we want more than two tiers, we have to consider more than one paradigm for the 3D IC.

¹This work has been done in the framework of EU-funded Horizon Europe Twinning project COIN-3D, under the Grant Agreement No. 101159667.

²https://github.com/coin3d-project/coin3d-project/tree/main/3DPX_testcase

For the F2F paradigm, after the topmost routing layer of the 2D layer stack, we have to reinsert the layer stack in reverse order. Of course, it is not reasonable to keep all the 2D stack layers for the 3D case, so we can choose up to which layer we want to preserve, depending on the routing congestion of the chip. To connect these two layer stacks, we have to create a mock via (MIV), which will connect to the topmost routing layers of each tier. To ensure the tool correctly interprets the connectivity, the VIA definitions and VIA RULES must be extended to describe the structure of the via and the spacing rules for the generation of via arrays. After that, the SPACING tables must be updated, adding the new top tier layers using the same values from the bottom tier layer respectively, and removing the unnecessary intermediate layers' information. Subsequently, the layer stack is ready to be used for the creation of a mock two-tier F2F 3D IC.

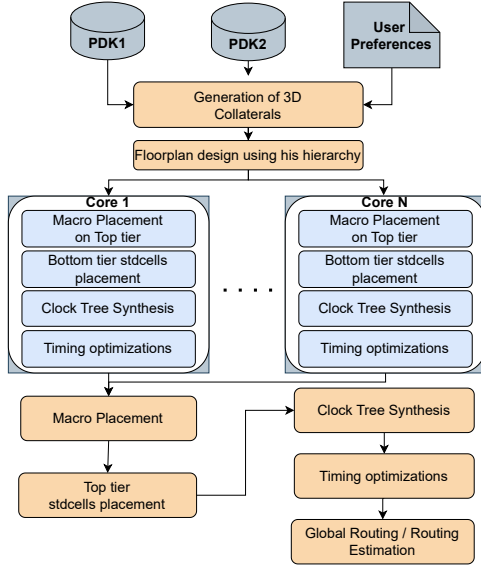


Fig. 1. The proposed 3DPX flow.

For the F2B integration approach, the two stacks will be concatenated without having the second inverted. In this approach, the introduced via connecting the tiers should have significantly larger size and spacings in order to simulate a realistic TSV. Following this, the same operations should be performed for the spacing, the VIA definitions, and the VIA rules tables, without any further modifications. The way to modify the layer stack for the B2B integration can be easily extracted based on the previous two descriptions.

For the F2F method to function correctly with OpenROAD, two critical modifications must be made to the LEF files. The first is that if we keep the same number of layers in the two tiers, the direction of the topmost layers will be the same, causing the OpenROAD router to raise errors. To resolve this issue, one has to add the $n+1$ layer of the 2D layer stack to the top tier. The second is that, for the logic-on-logic approach, some standard cell pins will be on the M1 layer of the top tier, making it the last routing tier of the combined stack. Due to certain engineering assumptions of the OpenROAD router, in order to avoid errors, we have to add a mock layer named M0 for example, on top of M1. It is important to specify in

the OpenROAD input variables that the top routing layer is M1 and not M0 so that the router knows where to terminate.

The next step, regarding the PDK preparation, is to set up the standard cells and IP LEF and LIB files. For the monolithic 3D approach, all standard cells and IPs must be duplicated in both the LEF and LIB files, with modified names using the suffixes `_bottom_tier` and `_top_tier`. Then, for each cell/IP, have to modify the pins layers accordingly. For the P&R flow that will be presented later on, we will need a *shrunked* version of the standard cells and IPs. To create this version, we have to copy the LEF file into a new one. In this new file, all the cells will have the size of a SITE, and no pin position has to be altered, as we want them to be projected in the original coordinates. For this *shrunked* version, the LIB file will remain the same. It is handy to keep separate LEF and LIB files for each tier and version.

Since this flow is intended for design exploration, it is not strictly necessary to extend other PDK files at this stage. Nevertheless, the tech file (.tf) could be extended using the same approach as the layer stack LEF file. Also, considering that the platform we are working on is the OpenROAD, the layer RC rules could be extended using the same trend, but the accuracy of the extractor would have been notably poor.

2) *3D OpenROAD flow*: As presented in Figure 1, the flow starts with the partitioning and the floorplanning of the design. At this stage, the top-level floorplan establishes the relative positions and dimensions of manually defined hierarchical blocks or clusters, ensuring efficient communication between them and optimizing the use of area and routing resources. Up to this point, these steps are not automated. For the first stage of our research, we considered that it would be beneficial to exploit the hierarchy of the design and use the larger modules as floorplanning blocks. However, the user can partition the design into blocks using an external tool and then use the provided API to set the locations of these blocks. At this point, the top-level floorplan establishes the relative positions and dimensions of manually defined hierarchical blocks or clusters, ensuring efficient communication between them and optimizing the use of area and routing resources.

The next step in our flow is to place and route each hierarchy block. For each block, the user can choose between the two known layering methods, logic-on-logic or memory-on-logic. To work with the first one, the user has to place the macros of the design either manually or using an external tool and insert the positions using OpenROAD API, `place_macro`, enabling the `-allow_overlap` option. After that, has to place the standard cells of each tier. In order not to agnostically place the standard cells of each tier in our flow we use the *shrunked* LEF version of both the standard cells and hard macros for both tiers and perform a global placement step, enabling the timing-driven, low-density and routability-driven options. After the initial global placement step, the user will use the API to fix the cells of one tier into positions and turn them to `COVER` type. This way, the cells of the fixed tier will not be recognized by the placer as fixed position cells, allowing overlaps with the cells of the other tier. Then, the user will perform an incremental placement step using the *shrunked* cell and macros sizes for the fixed tier and the actual cell and macros sizes for the other one. For this incremental step user can enable the same options as before or some of them to reduce runtime. Then will undo the conversion of the fixed tier cells and do the opposite for the other tier.

If the user wishes to proceed with the latest layering methodology, memory-on-logic, the flow becomes simpler. First, will use the native macro placer provided by OpenROAD to place the macros in the top tier. Of course, it has the option to use an external tool instead and use the API to insert the coordinates of the macros. After that, the user has to use the actual sizes of standard cells and the shrunk sizes of macros to perform the global placement step using the timing driven and routability option but not necessarily the low density one, as the utilization of the lower tier will be high enough. Then, will legalize the cells using the detailed placement step to place the cells into the rows. In both methods after the cells are legalized, the IOs of the block have to be optimized using the integrated functionalities of OpenROAD. It is important to set the appropriate pin constraints, according to the estimated position of the block in the top-level design.

After all the cells and IOs are placed and optimized, the user has to fix the timing of this block. This way, will use the OpenROAD commands to estimate parasitics in post placement state and perform an initial static timing analysis, followed by the appropriate timing fixes. To minimize the changes that will be done in each hierarchy block when all of them will be glued back together in the top level design, regarding adding, removing or placing cells, the next step is to perform the clock tree synthesis for this block. Accordingly, the timing analysis will be more accurate, and all the necessary extra cells will be inserted before the flow returns to the top-level design. After the clock tree insertion, the user will perform once more a static timing analysis along with the timing fixes, so the block can have a close estimation about the timing, the placement utilization, and the routability. Next, the user can perform either global or detailed route for each block. However, this might increase the API complexity that will be needed for the merging of the blocks to the top level-design. With this step, the block level flow is over, and the block is ready to be merged in the top-level design. It is important to mention that for each block, the flow can work independently, providing a concurrent implementation advantage.

So far, all the selected or created hierarchy blocks should have their cells placed and an initial notion of routability and timing analysis. Now, all these blocks have to be merged to the top-level design. Each hierarchical block has a produced Verilog file, which includes all the extra buffers inserted by the clock tree synthesis and a DEF file encapsulating the placement of the cells in the tier. To merge each block to the top-level design, we concatenate the Verilog file of each block to the top-level design Verilog file and merge the information from each DEF with that of the top-level design.

The next step is to place the remaining cells and macros of the top-level design. Usually, these cells belong to the logic between the large hierarchical blocks of the design and will not be a notable number of cells. Because of that user should select the tier that will insert this logic according to the utilization of each tier. It is recommended to place these cells in the top tier, allowing more space in the bottom tier for better routing results. After that, the flow will follow the logic-on-logic flow as described before. The cells of the bottom tier will be turned into COVER in order not to affect the placement and legalization of the top tier cells, and then will do the opposite. Be advised that if the design is not prohibitively large, performing an incremental placement on each tier using all the cells of that tier will result in better routing.

The next stop in the flow is the insertion of the power grid. Power grid insertion is a critical step in the ASIC physical design flow, where a robust network of power (VDD) and ground (VSS) lines is created to deliver stable voltage to all parts of the chip. This grid is typically implemented as a mesh or ring structure spanning multiple metal layers, designed to handle the expected current load while minimizing IR drop and ensuring uniform power distribution. Proper power grid design is essential for preventing voltage fluctuations, meeting electromigration limits, and supporting overall chip reliability and performance. This is one of the most important and ambiguous steps in 3D IC design, as it is completely dependent on the selected integration methodology and because there are a lot of approaches to do it. Since this flow is developed based on the OpenROAD platform, the user can design and specify the power grid to fit their application.

After that, the next steps are similar to the ones presented before in the block level logic-on-logic flow. First, the clock network will be created focusing on the clock IO to each block ports distribution, and then the timing optimization of the entire chip step will be performed. Then, a placement optimization iteration will be performed for both the top and bottom tiers, including all cells, to make sure that the cell positions are legal and none of the placement restrictions have been violated. To do that will incorporate the shrunk and cover technique described before for each tier, respectively. The flow will then proceed to the routing step, where the actual sizes of cells for both tiers will be used. This step has two major parts. The first is the pin accessing algorithm, and the second is the global routing algorithm. As this presented flow is only for design exploration purposes, the objective is to identify routing congestion points, thus, there is no point in spending significantly more time doing detailed routing. OpenROAD platform provides two ways to analyze and visualize routing congestion points. The first one is to estimate the congestion using the RUDY algorithm or the more accurate global routing based analysis. Both of them are fairly accurate compared to an industry standard congestion analysis tool.

III. EXPERIMENTS

In this section, two examples will be provided to demonstrate the impact and capabilities of the 3DPX flow on actual designs. The selected cases are a dual core and a quad core rocket tile RISC-V synthesized in the widely accessible NANGATE45. All experiments were conducted on a Linux-based machine, comprising an Intel Xeon W7-3565X Processor with 68 threads, no GPU support, and 128GB of RAM.

Prior to presenting the results, we state our assumptions during the implementation. First, the floorplans, along with the placement of some top-level design macros, were manually created and introduced to OpenROAD using the provided API. Additionally, for the cores, we adopted the memory-on-logic integration method. This way, all the hard macros of the core will be positioned in the top tier, while all the other standard cells will be placed beneath them in the bottom tier. This integration method was not arbitrarily chosen, since the quality of transistors in the bottom tier is significantly better than that of the top tier [6]. Considering that in this step we are working with the more delicate parts of the design, which in our case are the cores of a multicore system, we have to ensure high frequency operation and minimal system variation.

We now detail the floorplanning and key characteristics of the designs. The two designs feature two and four cores, respectively, with forty-four small size RAMs each, and four large sized RAMs and twenty-one small sized RAMs outside of the cores, as depicted in Fig. 2.

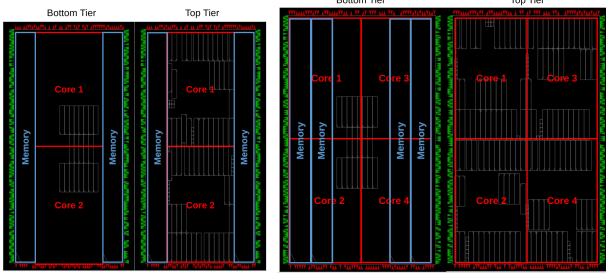


Fig. 2. The floorplanning of dual and quad core designs are presented across tiers. Within the RED rectangles, the memories and standard cells of the respective core will be placed. The BLUE and GREY rectangles identify the positions of the large and small size memories, respectively.

In these figures, the large size memories can be located on the left and right boundaries of the chip of the dual core design in both tiers, having a pair of large-size memories in each tier overlapping one another. For the quad core design, all the large sized memories are located in the bottom tier on the left and right boundaries of the chip, packed in pairs again. This floorplan was selected to increase the utilization of the bottom tier, as the ratio of standard cells and memories for this design is fairly small. The other twenty-one small sized memories, both in dual core and quad core are placed in the bottom tier at the center of the chip as they have the largest connectivity network with all the cores.

At this point, it is important to demonstrate that our flow's results and estimations are accurate and reflect reality. We validated the results of both designs against the 2D implementation using OpenROAD and compared the reported routing congestion estimation using an industrial standard tool. This approach ensures that the 3D integration results fully leverage the expected benefits described in the literature, while confirming that the algorithms employed by OpenROAD produce reliable, non arbitrary results.

One of the primary targets in the 3D integration is to significantly reduce the chip area consumption. Using our flow and manual floorplanning we managed to achieve almost 56% area reduction for the dual core design, from $6831230.1848\mu m^2$ in 2D to $2961800.0\mu m^2$ in 3D, and close to 53%, from $7944331.512\mu m^2$ in 2D to $3729860.0\mu m^2$ in 3D, for the quad core design. The important point here is that the produced 3D designs were able to meet the same timing constraints as the 2D integration while preserving routability. For reference, Fig. 3 presents the floorplans of the 3D and 2D implementations for the dual quad core designs, respectively.

In Fig. 4, the 3D implementation congestion estimation is compared against the industry standard tool. We can observe that the captured hotspot is similar between the two tools. However, because of the nature of the test case, the results might be characterized as ambiguous. To address this, we also provide the quad-core design. The corresponding results, included in the same figure, reinforce our claim that the RUDY congestion estimation algorithm is sufficiently accurate for the purposes of the presented flow.

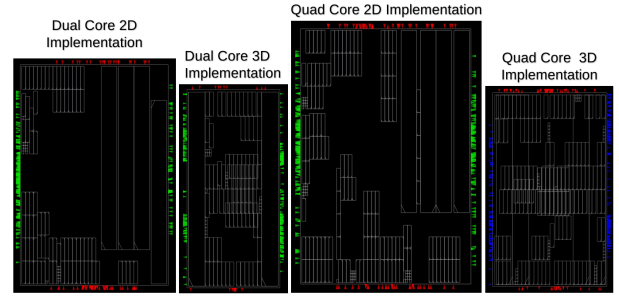


Fig. 3. From left to right, 2D and 3D implementations of the dual-core RISC-V design, followed by 2D and 3D implementations of the quad-core design.

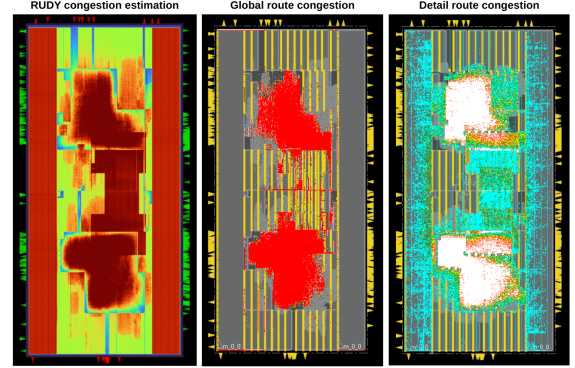


Fig. 4. Presents results for the dual-core design: OpenROAD's congestion estimation (left), industry-standard tool analysis post-global route (center), and post-route (right).

Finally, it is important to note that both 2D and 3D implementations met the same timing constraints, as with this flow the tool was able to perform clock tree synthesis and timing optimization across tiers. Additionally, the power network was effectively connected on both tiers, utilizing both the rings around the core and the stripes for each tier independently.

IV. CONCLUSIONS

In this paper, we presented 3DPX, an open-source flow for hierarchical design space exploration of 3D ICs based on OpenROAD, which supports a wide range of integration and stacking options. Experimental results on RISC-V multicore benchmarks demonstrate that 3DPX achieves 50% reduction in die area while being able to incorporate all available algorithms to optimize power and timing across tiers.

REFERENCES

- [1] K. Arabi, K. Samadi, and Y. Du, "3D VLSI: A scalable integration beyond 2D," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, p. 1–7.
- [2] Y. Xie and Y. Ma, "Design space exploration for 3D integrated circuits," in *2008 9th International Conference on Solid-State and Integrated-Circuit Technology*, 2008, pp. 2317–2320.
- [3] Y. Shi *et al.*, "Open3DBench: Open-source benchmark for 3D-IC backend implementation and PPA evaluation," 2025.
- [4] N. E. Bethur *et al.*, "Hier-3D: A methodology for physical hierarchy exploration of 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 7, pp. 1957–1970, 2024.
- [5] T. Ajayi *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proceedings of the 56th Annual Design Automation Conference*, 2019.
- [6] P. Batude *et al.*, "Advances, challenges and opportunities in 3D CMOS sequential integration," in *International Electron Devices Meeting*, 2011.